



# A Modified Decomposition Algorithm for Maximum Weight Bipartite Matching and Its Experimental Evaluation

Shibsankar Das

## ► To cite this version:

Shibsankar Das. A Modified Decomposition Algorithm for Maximum Weight Bipartite Matching and Its Experimental Evaluation . 2016. hal-01358220

**HAL Id: hal-01358220**

**<https://hal.science/hal-01358220>**

Preprint submitted on 31 Aug 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A Modified Decomposition Algorithm for Maximum Weight Bipartite Matching and Its Experimental Evaluation<sup>†</sup>

Shibsankar Das<sup>1,2‡</sup> and Rahul Kadyan<sup>2</sup>

<sup>1</sup>*Institute of Science, Banaras Hindu University, Varanasi - 221 005, India.*

<sup>2</sup>*Indian Institute of Technology Guwahati, Guwahati - 781 039, India.*

received 31<sup>st</sup> August 2016,

---

**Abstract:** Let  $G$  be an undirected bipartite graph with positive integer weights on the edges. We refine the existing decomposition theorem originally proposed by Kao et al., for computing maximum weight bipartite matching. We apply it to design an efficient version of the decomposition algorithm to compute the weight of a maximum weight bipartite matching of  $G$  in  $O(\sqrt{|V|}W'/k(|V|, W'/N))$ -time by employing an algorithm designed by Feder and Motwani as a subroutine, where  $|V|$  and  $N$  denote the number of nodes and the maximum edge weight of  $G$ , respectively and  $k(x, y) = \log x / \log(x^2/y)$ . The parameter  $W'$  is smaller than the total edge weight  $W$ , essentially when the largest edge weight differs by more than one from the second largest edge weight in the current working graph in any decomposition step of the algorithm. In best case  $W' = O(|E|)$  where  $|E|$  be the number of edges of  $G$  and in worst case  $W' = W$ , that is,  $|E| \leq W' \leq W$ . In addition, we talk about a scaling property of the algorithm and research a better bound of the parameter  $W'$ . An experimental evaluation on randomly generated data shows that the proposed improvement is significant in general.

**Keywords:** Graph algorithm, Weighted bipartite matching, Graph decomposition, Experimental evaluation, Random instances of weighted bipartite graph, Combinatorial optimization.

---

## 1 Introduction

Let  $G = (V = V_1 \cup V_2, E, Wt)$  be an undirected, weighted bipartite graph where  $V_1$  and  $V_2$  are two non-empty partitions of the vertex set  $V$  of  $G$ , and  $E$  is the edge set of  $G$  with positive integer weights on the edges which are given by the weight function  $Wt: E \rightarrow \mathbb{N}$ , where  $\mathbb{N}$  is the set of positive integers. Throughout the paper, we use the symbols  $N$  and  $W$  to denote the largest weight of any edge and the total weight of  $G$ , respectively. The weight of the graph  $G$  is defined by  $W = Wt(G) = \sum_{e \in E} Wt(e)$ . We also

---

<sup>†</sup>A preliminary version of this paper has been presented in the 11<sup>th</sup> International Conference on Theory and Applications of Models of Computation (TAMC 2014) [6]. The current expanded version includes a better bound of the parameter  $W'$  and the experimental evaluation of the theoretical claims made in previous version.

<sup>‡</sup>Corresponding author.

assume that the graph does not have any isolated vertex. For uniformity we treat an unweighted graph as a weighted graph having unit weight for all edges.

We use the notation  $\{u, v\}$  for an edge  $e \in E$  between  $u \in V_1$  and  $v \in V_2$ , and its weight is denoted by  $Wt(e) = Wt(u, v)$ . We also say that  $e = \{u, v\}$  is *incident* on vertices  $u$  and  $v$ ; and  $u$  and  $v$  are each *incident* with  $e$ . Two vertices  $u, v \in V$  of  $G$  are *adjacent* if there exists an edge  $e = \{u, v\} \in E$  of  $G$  to which they are both incident. Two edges  $e_1, e_2 \in E$  of  $G$  are *adjacent* if there exists a vertex  $v \in V$  to which they are both incident.

A subset  $M \subseteq E$  of edges is a *matching* if no two edges of  $M$  share a common vertex. A vertex  $v \in V$  is said to be *covered* or *matched* by the matching  $M$  if it is incident with an edge of  $M$ ; otherwise  $v$  is *unmatched* [2, 3].

A matching  $M$  of  $G$  is called a *maximum (cardinality) matching* if there does not exist any other matching of  $G$  with greater cardinality. We denote such a matching by  $mm(G)$ . The weight of a matching  $M$  is defined as  $Wt(M) = \sum_{e \in M} Wt(e)$ . A matching  $M$  of  $G$  is a *maximum weight matching*, denoted as  $mwm(G)$ , if  $Wt(M) \geq Wt(M')$  for every other matching  $M'$  of the graph  $G$ .

Observe that, if  $G$  is an unweighted graph then  $mwm(G)$  is a  $mm(G)$ , which we write as  $mwm(G) = mm(G)$  in short and its weight is given by  $Wt(mwm(G)) = |mm(G)|$ . Similarly, if  $G$  is an undirected and weighted graph with  $Wt(e) = c$  for all edges  $e$  in  $G$  and  $c$  is a constant then also we have  $mwm(G) = mm(G)$  with weight of the matching as  $Wt(mwm(G)) = c * |mm(G)|$ .

## 1.1 Our Contribution

In [18, 19], Kao et al. proposed a decomposition theorem and algorithm for computing weight of a Maximum Weight Bipartite Matching (MWBM) of the bipartite graph  $G$ . Our contribution in this paper is a revised version of the existing decomposition theorem and use it efficiently to design an improved version of the decomposition algorithm to estimate the weight of a MWBM of  $G$  in time  $O(\sqrt{|V|}W'/k(|V|, W'/N))$  by taking algorithm designed by Feder and Motwani [10] as base algorithm, where  $k(x, y) = \log x / \log(x^2/y)$ .

This algorithm bridges a gap between the best known time complexity of computing a Maximum Cardinality Matching (MCM) and that of computing a MWBM of a bipartite graph. In best case, computation of weight of a MWBM takes  $O(\sqrt{|V|}|E|/k(|V|, |E|))$  time which is the same as the complexity of the Feder and Motwani's algorithm [10] for computing MCM of unweighted bipartite graph; whereas in worst case it takes  $O(\sqrt{|V|}W/k(|V|, W/N))$ , that is,  $|E| \leq W' \leq W$ . Further, we provide an interesting scaling property of the algorithm and a better bound of the parameter  $W'$ . However, it seems to be a challenging problem to get rid of  $W$  or  $N$  from the complexity.

The modified algorithm works well for general  $W$ , but is best known for  $W' = o(|E| \log(|V|N))$ . We also design a revised algorithm to construct minimum weight cover of a bipartite graph in time  $O(\sqrt{|V|}W'/k(|V|, W'/N))$  to identify the edges involved in maximum weight bipartite matching. It is also possible to use other algorithms as a subroutine, for example, algorithms given by Hopcroft and Karp [16] and Alt et al. [1] in which case the running times of our algorithm will be  $O(\sqrt{|V|}W')$  and  $O((|V|/\log |V|)^{1/2}W')$ , respectively. An experimental evaluation on randomly generated bipartite graphs shows that the proposed improvement is significant in general.

## 1.2 Roadmap

In Section 2, we give a detailed summary of existing maximum matching algorithms and their complexities for unweighted and weighted bipartite graphs. Section 3 describes modified decomposition theorem

and an algorithm to compute the weight of a MWBM. The complexity analysis of the algorithm is discussed in Section 4. The algorithm to compute minimum weight cover of a bipartite graph is given in Section 5, which is used to find the edges of a MWBM. Section 6 provides the experimental comparisons between the modified algorithm and Kao et al.'s algorithm for randomly generated bipartite graphs. We summarize the results in Section 7.

## 2 Survey of Maximum Matching in Bipartite Graph

The problem of computing maximum matching in a given graph is one of the fundamental algorithmic problem that has played an important role in the development of combinatorial optimization and algorithmics. A survey of some of the well known existing maximum (cardinality) matching and maximum weight matching algorithms for bipartite graph are summarized in Table 1 and Table 2, respectively. The algorithms with best asymptotic bound are indicated by “\*” in these tables. A more detailed and technical discussion of the algorithms can be found in textbooks [20, 24, 25].

### 2.1 Maximum Cardinality Matching

For unweighted bipartite graphs, Hopcroft-Karp [16] algorithm, which is based on augmenting path technique, offers the best known performance for finding maximum matching in time  $O(|E|\sqrt{|V|})$ . In case of dense unweighted bipartite graphs, that is with  $|E| = \Theta(|V|^2)$ , slightly better algorithms exist. An algorithm by Alt et al. [1] obtains a maximum matching in  $O(|V|^{1.5}\sqrt{|E|/\log|V|})$  time. In case of  $|E| = \Theta(|V|^2)$ , this becomes  $O(|E|\sqrt{|V|/\log|V|})$  and is also  $\sqrt{\log|V|}$ -factor faster than Hopcroft-Karp algorithm. This speed up is obtained by an application of the fast adjacency matrix scanning technique of Cheriyan, Hagerup and Mehlhorn [4]. The algorithm proposed by Feder-Motwani [10] has the time complexity  $O(|E|\sqrt{|V|/k(|V|, |E|)})$ , where  $k(x, y) = \log x / \log(x^2/y)$ .

**Tab. 1:** Complexity survey of maximum unweighted bipartite matching algorithms.

Year	Author(s)	Complexity
1973 *	Hopcroft and Karp [16]	$O( E \sqrt{ V })$
1991	Alt, Blum, Mehlhorn and Paul [1]	$O( V ^{1.5}\sqrt{ E /\log V })$
1995 *	Feder and Motwani [10]	$O( E \sqrt{ V /k( V ,  E )})$

### 2.2 Maximum Weight Bipartite Matching

Several algorithms have also been proposed for computing maximum weight bipartite matching, improving both theoretical and practical running times. The well known Hungarian method, the first polynomial time algorithm, was introduced by Kuhn [21] and Munkres [22]. Fredman and Tarjan [11] improved this with running time  $O(|V|(|E| + |V|\log|V|))$  for sparse graph by using Fibonacci heaps. An  $O(|V|^{3/4}|E|\log N)$ -time scaling algorithm was proposed by Gabow [12] under the assumption that edge weights are integers. A different and faster scaling algorithm was given by Gabow and Tarjan [13] with running time  $O(\sqrt{|V|}|E|\log(|V|N))$ . Kao et al. [19] proposed an  $O(\sqrt{|V|}W/k(|V|, W/N))$ -time decomposition technique under the assumptions that weights on the edges are positive and  $W = o(|E|\log(|V|N))$ .

In addition to the above exact algorithms, several randomized and approximate algorithms are also proposed, see for example [9, 23].

**Tab. 2:** Complexity survey of maximum weight bipartite matching algorithms.

Year(s)	Author(s)	Complexity
1955, 1957	Kuhn [21], Munkres [22]	$O( V ^4)$ (Hungarian method)
1960	Iri [17, 24]	$O( V ^2 E )$
1969	Dinic and Kronrod [8, 24]	$O( V ^3)$
1984, 1987 *	Fredman and Tarjan [11]	$O( V ( E  +  V  \log  V ))$
1985	Gabow [12]	$O( V ^{3/4} E  \log N)$
1989 *	Gabow and Tarjan [13]	$O(\sqrt{ V } E  \log( V N))$
1999	Kao, Lam, Sung and Ting [18]	$O(\sqrt{ V }W)$
2001 *	Kao, Lam, Sung and Ting [19]	$O(\sqrt{ V }W/k( V , W/N))$
2014 * (This work)		$O(\sqrt{ V }W')$
		$O(( V /\log  V )^{1/2}W')$
		$O(\sqrt{ V }W'/k( V , W'/N))$

### 3 Refined Decomposition Theorem for Maximum Weight Bipartite Matching

We now propose a modified decomposition theorem which is a generalization of the existing decomposition theorem originally proposed by Kao et al. [18, 19] and use it to develop a revised version of the decomposition algorithm to decrease the number of iterations and speed up the computation of the weight of a MWBM. Let  $G = (V = V_1 \cup V_2, E, Wt)$  be an undirected, weighted bipartite graph 1 having  $V_1$  and  $V_2$  as partition of vertex set  $V$ . Further, let  $E = \{e_1, e_2, \dots, e_{|E|}\}$  be set of edges of  $G$  with weights  $Wt(e_i) = w_i$  for  $1 \leq i \leq |E|$ , where  $w_1, w_2, \dots, w_{|E|}$  are not necessarily distinct. As defined earlier, let  $N$  be the maximum edge weight, that is, for all  $i \in \{1, 2, \dots, |E|\}$ ,  $0 \leq w_i \leq N$ , and  $W = \sum_{1 \leq i \leq |E|} w_i$  be the total weight of  $G$ .

Our algorithm considers several intermediate graphs with lighter edge weights. During this process it is possible that weights of some of the edges may become zero. An edge  $e \in E$  is said to be *active* if its weight  $Wt(e) > 0$ , otherwise it is said to be *inactive*, that is when  $Wt(e) = 0$ . Let there be  $m' (\leq |E|)$  distinct edge weights in current working graph where  $w_1 < w_2 < \dots < w_{m'-1} < w_{m'}$ . We denote the first two distinct maximum edge weights in current working graph by  $H_1$  and  $H_2 (< H_1)$ , respectively. Assign  $H_2 = 0$  in case  $m' = 1$ .

We first build two new graphs referred to as  $G_h$  and  $G_h^\Delta$  from a given weighted bipartite graph  $G$ . For any integer  $h \in [1, N]$ , we decompose the graph  $G$  into two lighter weighted bipartite graph  $G_h$  and  $G_h^\Delta$  as proposed by Kao et al. [18, 19]. A minimum weight cover is a dual of maximum weight matching [19]. A *cover* of  $G$  is a function  $C: V_1 \cup V_2 \rightarrow \mathbb{N}_0$  such that  $C(v_1) + C(v_2) \geq Wt(v_1, v_2) \forall v_1 \in V_1$  and  $v_2 \in V_2$ . Let  $Wt(C) = \sum_{x \in V_1 \cup V_2} C(x)$ . A cover  $C$  is *minimum weight cover* if  $Wt(C)$  is minimum.

**Formation of  $G_h$  from  $G$ :** The graph  $G_h$  is formed by including those edges  $\{u, v\}$  of  $G$  whose weights

$Wt(u, v)$  lie in the range  $[N - h + 1, N]$ . Each edge  $\{u, v\}$  in graph  $G_h$  is assigned weight  $Wt(u, v) - (N - h)$ . For illustration,  $G_1$  is constructed by the maximum weight edges of  $G$  and assigned unit weight to each edge.

**Formation of  $G_h^\Delta$  from  $G$ :** Let  $C_h$  be the minimum weight cover of  $G_h$ . The graph  $G_h^\Delta$  is formed by including every edge  $\{u, v\}$  of  $G$  whose weight satisfies the condition

$$Wt(u, v) - C_h(u) - C_h(v) > 0.$$

The weight assigned to such an edge is  $Wt(u, v) - C_h(u) - C_h(v)$ .

**Theorem 3.1 (The Decomposition Theorem [19])** *Let  $G$  be an undirected, weighted bipartite graph. Then*

(a) *for any integer  $h \in [1, N]$ ,*

$$Wt(mwm(G)) = Wt(mwm(G_h)) + Wt(mwm(G_h^\Delta)),$$

(b) *in particular (trivial), for  $h = 1$ ,*

$$Wt(mwm(G)) = Wt(mm(G_1)) + Wt(mwm(G_1^\Delta)).$$

Note that the Theorem 3.1(b) is derived from Theorem 3.1(a), since for  $h = 1$ , we have

$$mwm(G_1) = mm(G_1)$$

and

$$Wt(mwm(G_1)) = Wt(mm(G_1)) = |mm(G_1)|.$$

The Theorem 3.1(b) is used recursively in the Algorithm 1 [19], to compute the weight of a maximum weight matching of the graph  $G$ .

---

**Algorithm 1** Kao et al.'s algorithm [19] to compute weight of a MWBM.

---

*Input:* A weighted, undirected, complete bipartite graph  $G$  with positive integer weights on the edges.

*Output:* Weight of a maximum weight matching of  $G$ , that is,  $Wt(mwm(G))$ .

Compute-MWM( $G$ )

- 1: Construct  $G_1$  from  $G$ .
  - 2: Compute  $mm(G_1)$  and find a minimum weight cover  $C_1$  of  $G_1$ .
  - 3: Construct  $G_1^\Delta$  from  $G$  and  $C_1$ .
  - 4: **if**  $G_1^\Delta$  is empty,
  - 5:     **then return**  $Wt(mm(G_1))$ ;
  - 6: **else return**  $Wt(mm(G_1)) + \text{Compute-MWM}(G_1^\Delta)$ .
-

**Remark 3.2** A graph  $G$  may not have all edge weights distinct. Consider the set of distinct edge weights of  $G$ . The Algorithm 1 works efficiently only when the largest edge weight differs by exactly one from the second largest edge weight of the current graph during an invocation of Theorem 3.1(b) in each iteration.

**Remark 3.3** Observe that for arbitrary  $h \in [1, N]$ ,  $mwm(G_h)$  need not be equal to  $mm(G_h)$ , that is, we cannot always conclude that  $mwm(G_h) = mm(G_h)$ .

One of our objectives is to investigate those values of  $h$  for which  $mwm(G_h)$  is equal to  $mm(G_h)$  apart from the trivial value of  $h$  as 1 in each iteration of the Algorithm 1 to generate  $G_h$  having all its edge weights as 1.

In order to get the speed up whenever possible, by decreasing the number of iterations whenever possible, we revise the Theorem 3.1(b) and propose Theorem 3.4 which gives a domain of  $h \in [1, N]$  where  $mwm(G_h) = mm(G_h)$  and as a consequence of that we can write

$$Wt(mwm(G_h)) = Wt(mm(G_h)) = h * |mm(G_h)|.$$

It works for  $h = 1$  and performs well especially when the largest edge weight differs by more than one from the second largest edge weight in the current graph in a decomposition step during an iteration.

**Theorem 3.4 (The Modified Decomposition Theorem)** The following equalities hold for any integer  $h \in [1, H_1 - H_2]$  where  $H_1$  and  $H_2$  ( $< H_1$ ) are the first two distinct maximum edge weights of graph  $G$ , respectively. We assign  $H_2 = 0$  in case all edge weights are equal.

- (a)  $mwm(G_h) = mm(G_h)$ ,
- (b)  $Wt(mwm(G)) = h * Wt(mm(G_h)) + Wt(mwm(G_h^\Delta))$ .

**Proof:** The proof of the above statements are based on the construction of new graphs  $G_h$  and  $G_h^\Delta$  from  $G$  and Theorem 3.1(a).

- (a) To prove that for any integer  $h$  where  $1 \leq h \leq H_1 - H_2$ ,  $mwm(G_h) = mm(G_h)$  holds true, it is enough to prove the same for the maximum value<sup>(i)</sup> of  $h$ , that is, for  $h = H_1 - H_2$ . As specified earlier, the construction of  $G_h$  is done by choosing those edges  $\{u, v\}$  of  $G$  that have weight

$$Wt(u, v) \in [N - h + 1, N] = [H_1 - (H_1 - H_2) + 1, H_1] = [H_2 + 1, H_1].$$

Since  $H_1 \in [H_2 + 1, H_1]$ ,  $G_h$  has only the heaviest edges of  $G$  and each such edge is assigned the same weight. Thus,  $mwm(G_h) = mm(G_h)$  for  $h = H_1 - H_2$ .

- (b) Observe that  $h \in [1, H_1 - H_2]$  and  $[1, H_1 - H_2] \subseteq [1, N]$ . So, by using Theorem 3.1(a) we have,  $\forall h \in [1, H_1 - H_2]$ ,

$$Wt(mwm(G)) = Wt(mwm(G_h)) + Wt(mwm(G_h^\Delta)).$$

---

<sup>(i)</sup> For illustration, consider  $h = c$  where  $1 \leq c \leq H_1 - H_2$ . Then as per the formation of  $G_h$  from  $G$ ,  $G_c$  is built by choosing those edges of  $G$  that have weight  $Wt(u, v) \in [N - (c - 1), N]$ . Since,  $c - 1 \geq 0$  and  $N \in [N - (c - 1), N]$  for any  $c \in [1, H_1 - H_2]$ ,  $G_c$  has only the heaviest edges of  $G$ . For optimization, choose  $h = H_1 - H_2$ , the maximum possible value of  $h$ .

Also by using the Theorem 3.4(a),  $mwm(G_h) = mm(G_h)$  for all  $h \in [1, H_1 - H_2]$ . Weight of each edge<sup>(ii)</sup>  $\{u, v\}$  in  $G_h$  is exactly  $Wt(u, v) - (N - h) = H_1 - (H_1 - h) = h$ . Therefore,

$$Wt(mwm(G_h)) = h * Wt(mm(G_h)) = h * |mm(G_h)|.$$

Hence for any integer  $h \in [1, H_1 - H_2]$ ,

$$\begin{aligned} Wt(mwm(G)) &= Wt(mwm(G_h)) + Wt(mwm(G_h^\Delta)) \\ &= h * Wt(mm(G_h)) + Wt(mwm(G_h^\Delta)). \end{aligned}$$

This completes the proof.  $\square$

**Remark 3.5** The equality  $mwm(G_h) = mm(G_h)$  in Theorem 3.4(a) is not true for  $h > H_1 - H_2$  and  $h \leq N$ .

To show that for any  $h \in [H_1 - H_2 + 1, N]$  the statement  $mwm(G_h) = mm(G_h)$  is not true, it is enough to show the same essentially for  $h = H_1 - H_2 + 1$ . Observe that  $h = H_1 - H_2 + 1 \geq 2$ , since  $H_1 > H_2$ . According to the construction of  $G_h$ , it is formed by edges  $\{u, v\}$  of  $G$  whose weights  $Wt(u, v) \in [N - h + 1, N] = [H_1 - (H_1 - H_2 + 1) + 1, H_1] = [H_2, H_1]$ , that is,  $G_h$  is built with the maximum weight edges and second maximum weight edges of  $G$ , because  $\{H_1, H_2\} \in [H_2, H_1]$ . The weight of each heaviest edge  $\{u, v\}$  of  $G$  in  $G_h$  is exactly

$$Wt(u, v) - (N - h) = H_1 - (H_1 - h) = h$$

which is greater than or equal to 2 and that of each second heaviest edge  $\{u, v\}$  of  $G$  in  $G_h$  is exactly

$$Wt(u, v) - (N - h) = H_2 - (H_1 - h) = (H_2 - H_1) + h = (1 - h) + h = 1.$$

Hence  $mwm(G_h) \neq mm(G_h)$  for such a value of  $h$ .

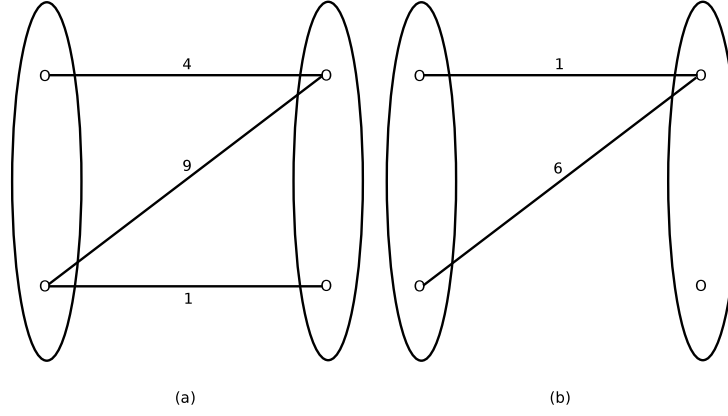
**Example 3.6** Consider the graph shown in the Figure 1(a). Let  $h = H_1 - H_2 + 1$ . So,  $h = H_1 - H_2 + 1 = 9 - 4 + 1 = 6$ . As shown in the Figure 1(b),  $G_h$  is formed by the edges  $\{u, v\}$  whose weights  $Wt(u, v) \in [N - h + 1, N] = [9 - 6 + 1, 9] = [4, 9]$  and their respective calculated weights are 6 and 1. Hence  $mwm(G_h) \neq mm(G_h)$ .

We use the modified decomposition Theorem 3.4 to design a recursive Algorithm 2 to compute the weight of a  $mwm(G)$ .

---

<sup>(ii)</sup> Only maximum weight edges of  $G$  are included in  $G_h$ .





**Fig. 1:** (a) An undirected bipartite graph  $G$  with positive integer weights on the edges. (b) Considering  $h = H_1 - H_2 + 1 = 6$ ,  $G_h$  is extracted, but  $mwm(G_h) \neq mm(G_h)$ .

---

**Algorithm 2** Compute weight of a maximum weight matching of  $G$ .

---

*Input:* A weighted, undirected, complete bipartite graph  $G$  with positive integer weights on the edges.

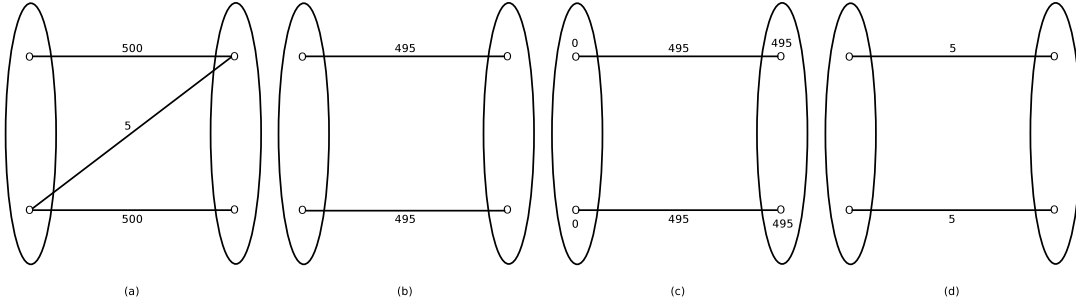
*Output:* Weight of a maximum weight matching of  $G$ , that is,  $Wt(mwm(G))$ .

WT-MWBM( $G$ )

- 1: Assume that initially  $Wt(mwm(G)) = 0$ .
  - 2: Find  $h = H_1 - H_2$  from the current working graph  $G$ .
  - 3: Construct  $G_h$  from  $G$ .
  - 4: Compute  $mm(G_h)$ .
  - 5: Find minimum weight cover  $C_h$  of  $G_h$ .
  - 6: Construct  $G_h^\Delta$  from  $G$  and  $C_h$ .
  - 7: **if**  $G_h^\Delta$  is empty (that is,  $G_h^\Delta$  has no active edge)
  - 8:   **then return**  $h * |mm(G_h)|$ ;
  - 9: **else return**  $h * |mm(G_h)| + \text{WT-MWBM}(G_h^\Delta)$ .
- 

**Example 3.7** Consider the bipartite graph shown in Figure 2(a). The Algorithm 2 finds the weight of a MWBM in just two iterations, as the algorithm is designed for the best  $h$  in every invocation of WT-MWBM(), whereas algorithm by Kao et al. [19] requires 500 iterations because it considers  $h = 1$  in every invocation of Compute-MWM().

Correctness of the algorithm follows from the construction of  $G_h$  and  $G_h^\Delta$  and the modified decomposition Theorem 3.4.



**Fig. 2:** (a) An undirected, weighted bipartite graph  $G$  with positive integer weights on the edges. In the current graph  $G$ ,  $h = 495$ . (b)  $G_h$  is extracted. (c)  $C_h$  is the weighted cover of  $G_h$ . (d)  $G_h^\Delta$  is formed from  $G_h$  and  $C_h$ . Compute  $\text{WT-MWBM}(G_h^\Delta)$ .

## 4 Complexity of the Modified Algorithm

Let  $G = (V = V_1 \cup V_2, E, W)$  be the initial input graph and  $N$  denotes the maximum edge weight of  $G$ , that is, for all  $i \in \{1, 2, \dots, |E|\}$ ,  $0 \leq w_i \leq N$  and  $W = \sum_{1 \leq i \leq |E|} w_i$  is the total weight of  $G$ . Further, let  $\{w_1, \dots, w_{m'}\}$  be the set of distinct edge weights of  $G$ , where  $m' \leq |E|$ .

Based on the constructions of  $G_h$  and  $G_h^\Delta$ , the modified decomposition Theorem 3.4 and the Algorithm 2, we can easily observe that in worst case the maximum number of possible iterations of  $\text{WT-MWBM}()$  is  $N$ , when  $h = 1$  in each iteration in the current working graph. Whereas in the best case, all the edge weights of  $G$  are the same and so we will have  $h = N$  for the present decomposition. As a consequence the algorithm will terminate in the first iteration itself.

As the complexity analysis of the Algorithm 2 is almost similar to that presented elsewhere [19], the details are available in Appendix 8 (see page 22). The algorithm takes  $O(\sqrt{|V|W'/k(|V|, W'/N)})$  time to compute the weight of a  $\text{mwm}(G)$  by using the algorithm by Feder and Motwani [10], as a subroutine.

Let  $L_i$  consists of edges of remaining  $G$  (after  $i - 1$ -th iteration) whose weights reduce in  $G_h^\Delta$  in  $i$ -th iteration. Also let there be  $p$  iterations,  $l_i = |L_i|$  for  $i = 1, 2, \dots, p \leq N$  and  $h_i = H_{i1} - H_{i2}$  in the  $i$ -th iteration, where  $H_{i1}$  and  $H_{i2}$  ( $< H_{i1}$ ) are the first two distinct maximum edge weights of the remaining graph  $G$  after the  $i - 1$ -th iteration.

From the detailed complexity analysis we have,  $l_1 h_1 + l_2 h_2 + \dots + l_p h_p = W$ . Let  $l_1 + l_2 + \dots + l_p = W'$ . Observe that, in worst case, if  $h_i = 1$  for all  $i \in [1, p]$ , then  $W' = \sum_{i=1}^p l_i = W$ . And in best case, if  $h_1 = N$ , then  $W' = |E|$ . Moreover, the parameter  $W'$  is smaller than  $W$ , essentially when the largest edge weight differs by more than one from the second largest edge weight in the current working graph in decomposition step during at least one iteration of the algorithm. Therefore in best case<sup>(iii)</sup>, it requires  $O(\sqrt{|V||E|/k(|V|, |E|)})$  time and in worst case  $O(\sqrt{|V|W/k(|V|, W/N)})$ , to compute weight of a maximum weight matching. That is,  $|E| \leq W' \leq W$ .

This time complexity bridges a gap between the best known time complexity for computing a Maximum Cardinality Matching (MCM) of unweighted bipartite graph and that of computing a MWBM of a

<sup>(iii)</sup> In best case, all the edge weights of  $G$  are the same. So, the algorithm terminates in just one iteration and hence  $W' = O(|E|)$ .

weighted bipartite graph. In best case, for computation of weight of a MWBM, the Algorithm 2 takes  $O(\sqrt{|V||E|}/k(|V|, |E|))$  time which is the same as the complexity of the Feder and Motwani's algorithm [10] for computing MCM of unweighted bipartite graph; whereas in worst case it (Algorithm 2) takes  $O(\sqrt{|V|W}/k(|V|, W/N))$  time which is the same as the complexity of the Kao et al.'s algorithm [19]. However, it is very difficult and challenging to get rid of  $W$  or  $N$  from the complexity. This modified algorithm works well for general  $W$ , but is best known for  $W' = o(|E| \log(|V|N))$ .

#### 4.1 Some More Advantages: Scaling Up and Down, and GCD Properties

Some other advantages of the modified decomposition algorithm is stated by the following propositions. Let  $G = (V, E, Wt)$  be an undirected, weighted bipartite graph,  $E = \{e_1, e_2, \dots, e_{|E|}\}$  be the set of positive integer weight edges with weights  $Wt(e_i) = w_i > 0$  (where  $1 \leq i \leq |E|$ ),  $N$  be the maximum edge weight and  $W = \sum_{1 \leq i \leq |E|} w_i$  be the total weight of  $G$ . The modified decomposition Algorithm 2 computes weight of a maximum weight bipartite matching of  $G$  in  $O(\sqrt{|V|W'}/k(|V|, W'/N))$  time, where  $|E| \leq W' \leq W$ .

**Proposition 4.1 (Multiplicative Scaling Up Property)** *Let  $\hat{G}$  be a new weighted bipartite graph constructed by multiplying a large constant  $\alpha \in \mathbb{N}$  to each edge weight  $w_i$  of the initial weighted bipartite graph  $G$ . Then for both the graphs  $G$  and  $\hat{G}$ , the complexity of the Algorithm 2 remains  $O(\sqrt{|V|W'}/k(|V|, W'/N))$  where  $|E| \leq W' \leq W$ ; whereas for the graph  $\hat{G}$ , the complexity of the Algorithm 1 becomes  $O(\sqrt{|V|\alpha W}/k(|V|, \alpha W/N))$ .*

**Proof:** As mentioned in the detailed complexity analysis of the Algorithm 2 (described in Appendix 8, page 22), let  $L_i$  consists of edges of remaining graph  $G$  (left after  $i - 1$ -th iteration), whose weights reduce in  $G_h^\Delta$  in the  $i$ -th iteration of WT-MWBM(). Assume that there be  $p$  iterations for the Algorithm 2,  $l_i = |L_i|$  for  $i = 1, 2, \dots, p \leq N$  and  $h_i = H_{i1} - H_{i2}$  in the  $i$ -th iteration, where  $H_{i1}$  and  $H_{i2} (< H_{i1})$  are the first two distinct maximum edge weights of the remaining graph  $G$  after  $i - 1$ -th iteration. From the detailed complexity analysis we have,

$$l_1 h_1 + l_2 h_2 + \dots + l_p h_p = W \quad \text{and} \quad l_1 + l_2 + \dots + l_p = W'.$$

Observe that for the new graph  $\hat{G}$ , the number of iterations in Algorithm WT-MWBM( $\hat{G}$ ) still remains  $p$  and in the computation of WT-MWBM( $\hat{G}$ ),  $L_i$  consists of  $l_i$  number of edges of the remaining graph  $\hat{G}$  (after  $i - 1$ -th iteration), whose weights reduce in  $\hat{G}_h^\Delta$  in  $i$ -th iteration of WT-MWBM(). In this case, if  $h'_i = H'_{i1} - H'_{i2}$  in the  $i$ -th iteration, where  $H'_{i1}$  and  $H'_{i2} (< H'_{i1})$  are the first two distinct maximum edge weights of the remaining graph  $\hat{G}$  after  $i - 1$ -th iteration, respectively, then

$$h'_i = H'_{i1} - H'_{i2} = \alpha * H_{i1} - \alpha * H_{i2} = \alpha h_i \quad \text{where} \quad i = 1, 2, \dots, p,$$

$$l_1 h'_1 + l_2 h'_2 + \dots + l_p h'_p = l_1 \alpha h_1 + l_2 \alpha h_2 + \dots + l_p \alpha h_p = \alpha W,$$

and

$$l_1 + l_2 + \dots + l_p = W'.$$

Therefore, the modified Algorithm 2 will take  $O(\sqrt{|V|W'}/k(|V|, W'/N))$  time to compute the weight of a  $mwm(\hat{G})$  by using the algorithm by Feder and Motwani [10] as a subroutine; whereas time required for the Kao et. al.'s Algorithm 1 is  $O(\sqrt{|V|\alpha W}/k(|V|, \alpha W/N))$  time.  $\square$

That is, multiplication by an integer constant to all the weight of edges of a weighted bipartite graph  $G$  does not affect the time complexity of the modified decomposition algorithm for computing the weight of a MWBM of the bipartite graph. The following remark talks about a conditional scaling down property of the algorithm for the graph  $G$ .

**Remark 4.2 (Multiplicative Scaling Down Property)** *Let we scale down each edge weights of  $G$  by multiplying a factor of  $\frac{1}{\alpha}$  and get a new graph  $\hat{G}$ , where  $\alpha$  is the Greatest Common Divisor (GCD) of the positive edge weights of  $G$ . Then the time complexity of the Algorithm 2 for computing a MWBM of both the graphs  $G$  and  $\hat{G}$  remains same.*

Though during the complexity calculation of Algorithm 2 we have stated a bound for  $W'$  as:  $|E| \leq W' \leq W$ , but the following proposition gives a more better bound of the parameter  $W'$ .

**Proposition 4.3 (GCD Property)** *Let  $G = (V, E, Wt)$  be an undirected, weighted bipartite graph and  $E = \{e_1, e_2, \dots, e_{|E|}\}$  be the set of positive weight edges with weights  $Wt(e_i) = w_i > 0$  for  $1 \leq i \leq |E|$ . Further, let the GCD of the positive edge weights of  $G$  is denoted by  $GCD(w_1, w_2, \dots, w_{|E|})$ , then*

$$|E| \leq W' \leq \frac{W}{GCD(w_1, w_2, \dots, w_{|E|})} \leq W.$$

**Proof:** Without going into more detailed and repeated writing, as mentioned in the previous Proposition 4.1, we have:

$$l_1 h_1 + l_2 h_2 + \dots + l_p h_p = W, \text{ where } h_i = H_{i1} - H_{i2} \text{ and } l_1 + l_2 + \dots + l_p = W'.$$

Let  $g = GCD(w_1, w_2, \dots, w_{|E|})$ . Observer that, in any iteration  $i$  (where  $i = 1, 2, \dots, p$ ) both  $H_{i1}$  and  $H_{i2}$  are divisible by  $g$ . Hence, according to the definition of  $h_i$ s, each  $h_i = H_{i1} - H_{i2}$  is also divisible by the factor  $g$ .

$$\begin{aligned} \text{Therefore, } W' &= l_1 + l_2 + \dots + l_p \\ &\leq l_1 (h_1/g) + l_2 (h_2/g) + \dots + l_p (h_p/g) \\ &\leq (l_1 h_1 + l_2 h_2 + \dots + l_p h_p)/g = \frac{W}{g} \leq W. \end{aligned}$$

This completes the proof. □

## 4.2 Complexity Analysis by Considering Other Base Algorithms

We also analyze the complexity of the Algorithm 2 by considering the Hopcroft-Karp algorithm [16] and Alt-Blum-Mehlhorn-Paul algorithm [1] as base algorithms.

**With Respect to the Hopcroft-Karp Algorithm:** Hopcroft-Karp algorithm [16] presents the best known worst-case performance for getting a maximum matching in a bipartite graph with runtime of  $O(\sqrt{|V|}|E|)$ . Hence the recurrence relation for running time of the Algorithm 2 with respect to Hopcroft-Karp algorithm is

$$\begin{aligned} T(|V|, W', N) &= O(\sqrt{|V|}l_1) + T(|V|, W'', N'') \\ \text{and } T(|V|, 0, 0) &= 0 \end{aligned}$$

$$\begin{aligned}
\text{Therefore, } T(|V|, W', N) &= O(\sqrt{|V|}l_1) + O(\sqrt{|V|}l_2) + \cdots + O(\sqrt{|V|}l_p) \\
&= O\left(\sqrt{|V|} \sum_{i=1}^p l_i\right) = O(\sqrt{|V|}W').
\end{aligned}$$

**With Respect to the Alt-Blum-Mehlhorn-Paul Algorithm:** A bit better algorithm for dense bipartite graph is Alt-Blum-Mehlhorn-Paul algorithm [1] which is  $(\log |V|)^{1/2}$ -factor faster than Hopcroft-Karp algorithm for maximum bipartite matching. Hence the time complexity, with respect to Alt-Blum-Mehlhorn-Paul algorithm as a base algorithm, is  $O((|V|/\log |V|)^{1/2}W')$  and it is  $(\log |V|)^{1/2}$ -factor faster than the above case.

## 5 Finding a Maximum Weight Matching

The Algorithm 2 computes only the weight of a  $mwm(G)$  of a given graph  $G$ . To find the edges of a  $mwm(G)$ , we give a revised algorithm for constructing a Minimum Weight Cover (MWC) of  $G$  which is a dual of maximum weight matching. As mentioned before, a *cover* of  $G$  is a function  $C: V_1 \cup V_2 \rightarrow \mathbb{N}_0$  such that  $C(v_1) + C(v_2) \geq Wt(v_1, v_2) \quad \forall v_1 \in V_1 \text{ and } v_2 \in V_2$ . Let  $Wt(C) = \sum_{x \in V_1 \cup V_2} C(x)$ . We say  $C$  is *minimum weight cover* if  $Wt(C)$  is minimum. Let  $C$  be a MWC of a graph  $G$ .

**Lemma 5.1 ([19])** *Let  $C_h^\Delta$  be any minimum weight cover of  $G_h^\Delta$ . If  $C$  is a function on  $V(G)$  such that for every  $u \in V(G)$ ,  $C(u) = C_h(u) + C_h^\Delta(u)$ , then  $C$  is minimum weight cover of  $G$ .*

Using this lemma we design an  $O(\sqrt{|V|}W'/k(|V|, W'/N))$ -time revised algorithm to compute a MWC of  $G$ . The correctness of this algorithm is clear from the Lemma 5.1 and the time complexity analysis is similar to that given in the previous section.

---

**Algorithm 3** Calculate a MWC  $C$  of  $G$ .

---

*Input:* A weighted, undirected, complete bipartite graph  $G$  with positive integer weights on the edges.

*Output:* A minimum weight cover  $C$  of  $G$ .

MWC( $G$ )

- 1: Assume that initially  $Wt(mwm(G)) = 0$ .
  - 2: Find  $h \leftarrow H_1 - H_2$  from the current working graph  $G$ .
  - 3: Construct  $G_h$  from  $G$ .
  - 4: Compute  $mm(G_h)$ .
  - 5: Find minimum weight cover  $C_h$  of  $G_h$ .
  - 6: Construct  $G_h^\Delta$  from  $G$  and  $C_h$ .
  - 7: **if**  $G_h^\Delta$  is empty (that is,  $G_h^\Delta$  has no active edge)
  - 8:     **then return**  $C_h$ ;
  - 9: **else**
  - 10:     $C_h^\Delta \leftarrow \text{MWC}(G_h^\Delta)$ ;
  - 11:    **return**  $C$ , where  $C(u) = C_h(u) + C_h^\Delta(u)$  for all nodes  $u$  in  $G$ .
- 

Now as deduced by Kao et al. in [19], finding a maximum weight matching by using the given vertex

cover takes  $O(\sqrt{|V|}|E|/k(|V|, |E|))$  time. Since  $|E| \leq W' \leq W$ , so altogether  $O(\sqrt{|V|}W'/k(|V|, W'/N))$  time requires to find a MWBM of  $G$ .

## 6 Experimental Evaluation

The Algorithm 2 is efficient because of the modified decomposition Theorem 3.4. In order to understand the practical importance of the Algorithm 2, we report experimental evaluations of the same for the randomly generated weighted bipartite graphs.

### 6.1 Implementation and Experimental Environments

We have implemented both Kao et al.'s algorithm [19] and Algorithm 2 in C++ and compiled them using g++ 4.8.2-19ubuntu1 compiler. All the experiments have been performed on a Desktop PC with an Intel® Xeon®(E5620 @ 2.40 GHz) Processor, 32.00 GB RAM and 1200 GB Hard Disk, running the Ubuntu 14.04.1 LTS (Trusty Tahr) 64-bit Operating System.

### 6.2 Input Data Description and Its Randomness

For a frame of fixed number of vertices in a partition of the vertex set and fixed weight of bipartite graph  $G$ , we have generated the random weighted  $G$  by assigning random (uniformly distributed) weight to the randomly (uniformly distributed) picked up edges of  $G$ . The outputs of these experiments for an input bipartite graph  $G$  are:

- (a) the number of iterations of WT-MWBM() and Compute-MWM() in the Algorithms 2 and 1, respectively, for the graph  $G$  and
- (b) total time taken by the respective algorithms to compute the weight of a MWBM of  $G$ .

As mentioned in [7, 15] the Approximate Parameterized String Matching (APSM) problem under Hamming distance error model is computationally equivalent to the MWBM problem in graph theory. The input data relation between the above problems are:

- (a) length of the pattern is equal to weight of the bipartite graph, and
- (b) alphabet size of the pattern is equal to number of vertices in a partition of the vertex set of the corresponding bipartite graph.

### 6.3 Experimental Results

We have tested the respective algorithms with large input data sets. The details are given below. In each of the graphs, the output of our Algorithm 2 (denoted in short by “Modified Algorithm”) corresponds to the red colored unbroken line, whereas that of for the Algorithm 1 (denoted in short by “Kao et al.’s Algorithm”) corresponds to the red colored dotted line.

**Experiment 6.1** *This experiment is done for a total of 250 pseudo-randomly generated bipartite graphs, each of its weight is fixed to 1000 unit where size of each of the partitions of the vertex set of bipartite graph varies from 2 to 26.*

*Each numerical row of the Table 3 is corresponding to 10 different random graphs, each of whose size of each partition of the vertex set and weight of the graphs are fixed. Only for this experimental result,*

**Tab. 3:** Efficiency comparison between the Algorithms 2 and 1 for the 250 pseudo-randomly generated weighted bipartite graphs as considered in Experiment 6.1.

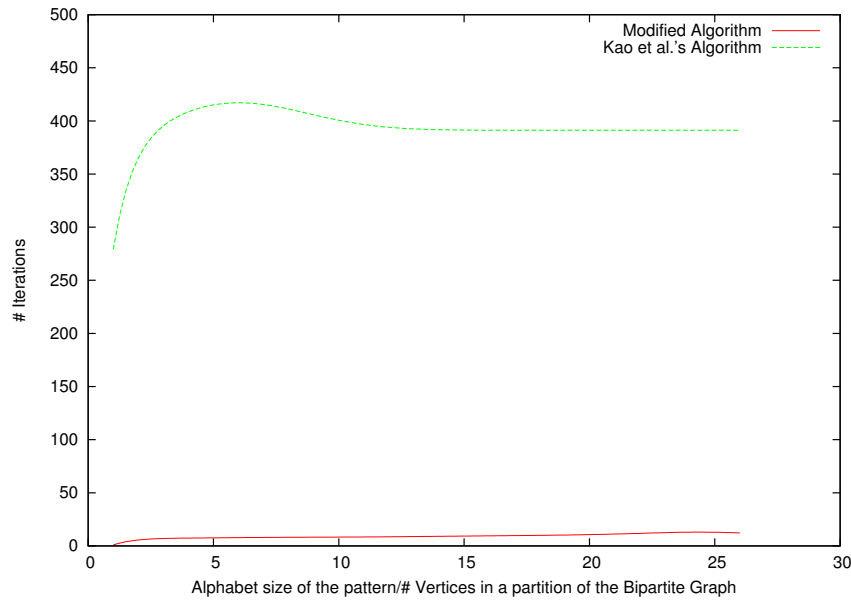
# Vertices in a Partition	Weight of Graph	Algorithm 2		Algorithm 1 (by Kao et al.)	
		# Iterations	Time (Sec.)	# Iterations	Time (Sec.)
2	1000	8.40	0.000050	422.60	0.002356
3	1000	8.20	0.000084	397.60	0.003769
4	1000	6.60	0.000104	402.20	0.005922
5	1000	7.20	0.000146	444.40	0.009566
6	1000	8.60	0.000214	408.60	0.010882
7	1000	7.00	0.000251	418.60	0.014451
8	1000	10.40	0.000405	455.20	0.019964
9	1000	7.20	0.000376	366.40	0.018605
10	1000	7.40	0.000411	400.20	0.024284
11	1000	8.60	0.000525	391.20	0.025592
12	1000	9.00	0.000670	391.20	0.031218
13	1000	8.40	0.000722	391.20	0.035947
14	1000	8.40	0.000811	391.20	0.039951
15	1000	10.20	0.001089	391.20	0.044228
16	1000	9.60	0.001156	391.20	0.048768
17	1000	9.40	0.001217	391.20	0.053661
18	1000	9.60	0.001411	391.20	0.058682
19	1000	11.00	0.001680	391.20	0.064456
20	1000	8.60	0.001492	391.20	0.070000
21	1000	11.80	0.002226	391.20	0.077403
22	1000	11.00	0.002284	391.20	0.083169
23	1000	13.40	0.002924	391.20	0.089702
24	1000	13.80	0.003254	391.20	0.097006
25	1000	13.00	0.003327	391.20	0.103550
26	1000	12.20	0.003342	391.20	0.110369

each row reports the average output of 10 different random graphs, each of whose number of vertices and weight are fixed.

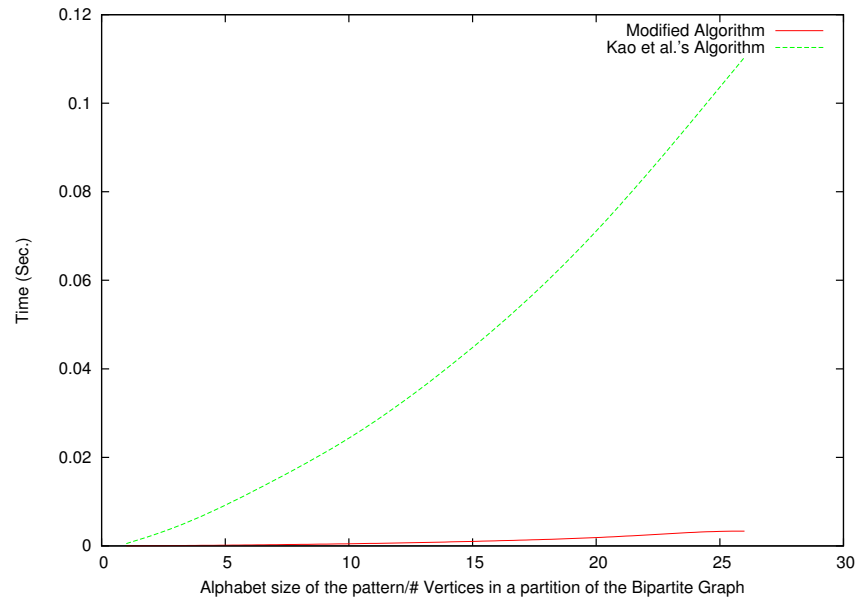
For example, the numerical row corresponding to ‘# Vertices in a partition’ equal to 15 reports the following. For the 10 randomly generated different bipartite graphs, each of whose size of the vertex set is 30 and weight is 1000 unit. And on an average the number of iterations of WT-MWBM( ) and Compute-MWM( ) in the Algorithms 2 and 1 are 10.20 and 391.20, respectively; whereas average time taken by the respective algorithms to compute the weight of a MWBM are 0.001089 and 0.044228 seconds.

Figure 3 shows the comparison on the number of iterations for the random graphs with different size partition of the vertices for the Algorithms 2 and 1. Similarly, Figure 4 gives the comparison time taken by the same algorithms for the random graphs with different size partition of the vertices.  $\square$

The next two experiments are done over the graphs corresponding to the randomly generated strings over the DNA alphabet  $\Sigma = \{A, C, G, T\}$  of different lengths.



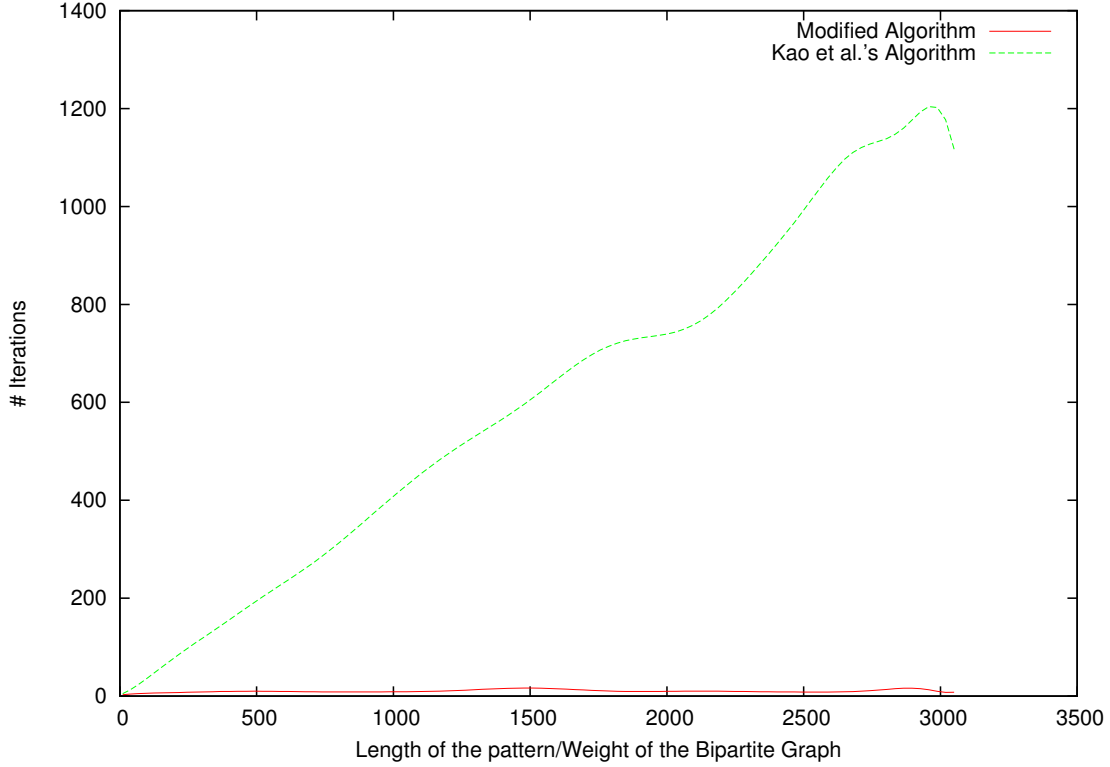
**Fig. 3:** Partition size vs. Iteration graph corresponding to the Experiment 6.1. Weight of each input graph is fixed to be 1000 unit.



**Fig. 4:** Partition size vs. Time graph corresponding to the Experiment 6.1. Weight of each input graph is fixed to be 1000 unit.



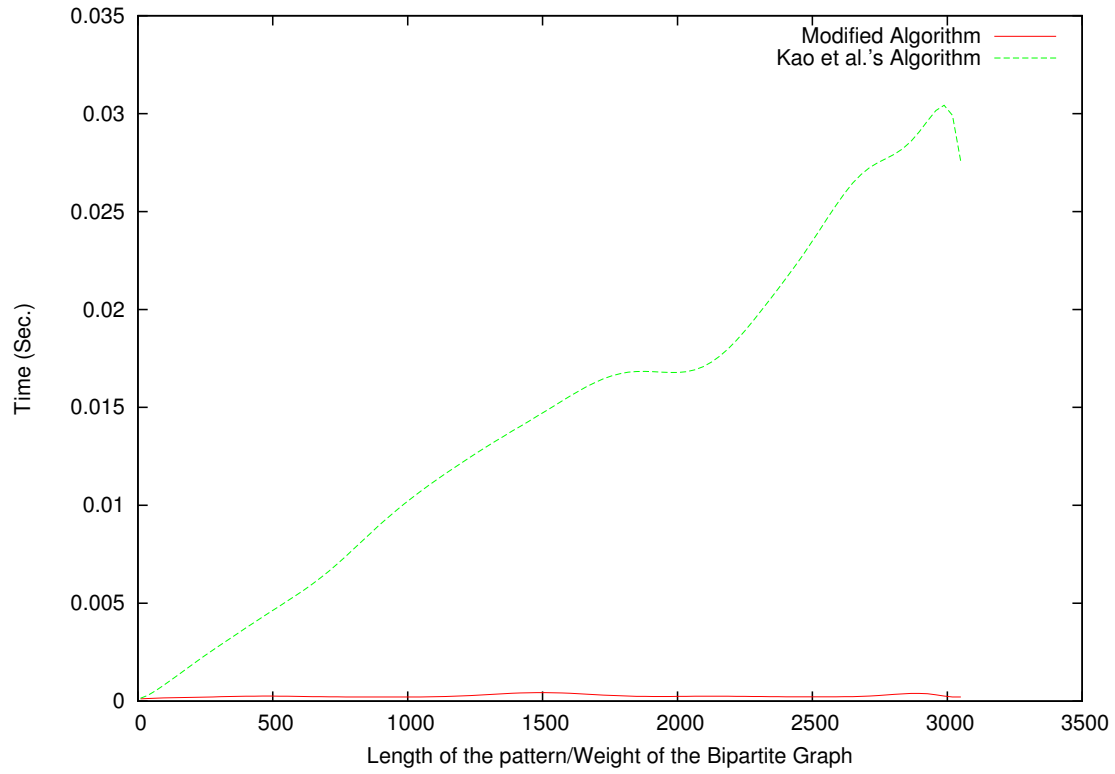
**Experiment 6.2** In this experiment we have fixed the size of each partition of each graph to 4 and randomly generated a total of 62 bipartite graphs for 62 different weights. See Table 4 for more details. Unlike previous experiment, each row reports the iterations and time comparison of the Algorithms 2 and 1 on a randomly generated bipartite graph with fixed size vertex and weight. Figures 5 and 6 describe the pictorial representation of the Table 4.  $\square$



**Fig. 5:** Weight vs. Iteration graph corresponding to the Experiment 6.2. The number of vertices in each partition of the vertex set is fixed to be 4.

**Tab. 4:** Experimental result for the 62 pseudo-randomly generated weighted bipartite graphs as considered in Experiment 6.2. The number of vertices in each partition of the vertex set of each of the graphs is fixed to be 4, but weight of the graph varies.

# Vertices in a Partition	Weight of Graph	Algorithm 2		Algorithm 1 (by Kao et al.)	
		# Iterations	Time (Sec.)	# Iterations	Time (Sec.)
4	10	3.00	0.000121	5.00	0.000152
4	50	4.00	0.000109	13.00	0.000229
4	100	9.00	0.000260	38.00	0.000965
4	150	6.00	0.000165	52.00	0.001257
4	200	5.00	0.000141	81.00	0.001641
4	250	5.00	0.000142	109.00	0.002703
4	300	18.00	0.000419	133.00	0.003255
4	350	5.00	0.000144	116.00	0.002648
4	400	6.00	0.000192	139.00	0.003666
4	450	4.00	0.000122	192.00	0.004288
4	500	31.00	0.000745	189.00	0.004863
4	550	6.00	0.000165	203.00	0.004828
4	600	6.00	0.000159	277.00	0.006564
4	650	6.00	0.000166	298.00	0.006679
4	700	9.00	0.000195	186.00	0.003468
4	750	8.00	0.000222	268.00	0.007218
4	800	7.00	0.000188	243.00	0.005828
4	850	8.00	0.000193	390.00	0.009581
4	900	8.00	0.000205	380.00	0.010258
4	950	11.00	0.000297	395.00	0.011187
4	1000	11.00	0.000248	368.00	0.009515
4	1050	12.00	0.000230	466.00	0.012499
4	1100	5.00	0.000135	535.00	0.012997
4	1150	6.00	0.000126	405.00	0.008078
4	1200	8.00	0.000205	397.00	0.008670
4	1250	6.00	0.000168	602.00	0.014516
4	1300	6.00	0.000168	648.00	0.016127
4	1350	13.00	0.000283	553.00	0.015214
4	1400	6.00	0.000158	639.00	0.016306
4	1450	10.00	0.000242	426.00	0.009439
4	1500	94.00	0.002554	456.00	0.011925
4	1550	9.00	0.000267	591.00	0.017001
4	1600	6.00	0.000162	775.00	0.016471
4	1650	7.00	0.000189	676.00	0.015674
4	1700	6.00	0.000162	665.00	0.013834
4	1750	6.00	0.000162	860.00	0.021617
4	1800	6.00	0.000163	701.00	0.017824
4	1850	6.00	0.000150	777.00	0.016954
4	1900	6.00	0.000142	827.00	0.019956
4	1950	8.00	0.000194	788.00	0.018098
4	2000	6.00	0.000168	690.00	0.016825
4	2050	6.00	0.000175	584.00	0.011463
4	2100	37.00	0.000864	727.00	0.016040
4	2150	5.00	0.000139	585.00	0.009586
4	2200	8.00	0.000186	747.00	0.016389
4	2250	6.00	0.000175	897.00	0.022177
4	2300	10.00	0.000252	1091.00	0.028346
4	2350	6.00	0.000164	740.00	0.014673
4	2400	8.00	0.000185	954.00	0.022879
4	2450	6.00	0.000168	998.00	0.023334
4	2500	18.00	0.000439	735.00	0.017273
4	2550	5.00	0.000159	1086.00	0.023379
4	2600	7.00	0.000200	1035.00	0.027333
4	2650	6.00	0.000155	1313.00	0.032712
4	2700	4.00	0.000133	1348.00	0.031344
4	2750	10.00	0.000254	922.00	0.020268
4	2800	9.00	0.000271	1030.00	0.028735
4	2850	15.00	0.000361	1206.00	0.029624
4	2900	42.00	0.000958	1144.00	0.026569
4	2950	5.00	0.000153	1266.00	0.030598
4	3000	6.00	0.000180	1249.00	0.033715
4	3050	8.00	0.000216	1117.00	0.027538



**Fig. 6:** Weight vs. Time graph corresponding to the Experiment 6.2. The number of vertices in each partition of the vertex set is fixed to be 4.

**Experiment 6.3** In the final experiment also we have fixed the size of a partition of each graph to 4 and but for a total of 71 randomly generated bipartite graphs for 71 different and large weights. See Table 5 for more details.  $\square$

**Tab. 5:** Experimental result for the 71 pseudo-randomly generated bipartite graphs as considered in Experiment 6.3. Cardinality of each partition of the vertex set is fixed to be 4, but weight of the graph varies largely.

# Vertices in a Partition	Weight of Graph	Algorithm 2		Algorithm 1 (by Kao et al.)	
		# Iterations	Time (Sec.)	# Iterations	Time (Sec.)
4	1000	11.00	0.000241	368.00	0.009156
4	10000	7.00	0.000196	4284.00	0.009156
4	20000	8.00	0.000193	7722.00	0.009156
4	30000	6.00	0.000161	14942.00	0.352380
4	40000	12.00	0.000326	19786.00	0.589580
4	50000	8.00	0.000213	22172.00	0.619935
4	60000	8.00	0.000207	28763.00	0.806167
4	70000	8.00	0.000215	22042.00	0.588712
4	80000	9.00	0.000190	28054.00	0.662195
4	90000	10.00	0.000240	21440.00	0.447115
4	100000	36.00	0.000970	32975.00	0.865868
4	110000	10.00	0.000222	53322.00	1.320364
4	120000	8.00	0.000199	57741.00	1.466250
4	130000	6.00	0.000156	54970.00	1.209947
4	140000	9.00	0.000204	50849.00	1.102937
4	150000	12.00	0.000221	65220.00	1.502660
4	160000	9.00	0.000183	44405.00	0.868532
4	170000	8.00	0.000188	71264.00	1.668321
4	180000	9.00	0.000212	68565.00	1.391208
4	190000	7.00	0.000182	58417.00	1.261169
4	200000	8.00	0.000199	84882.00	2.177765
4	210000	21.00	0.000411	55160.00	0.861068
4	220000	7.00	0.000171	90913.00	1.931699
4	230000	10.00	0.000247	87201.00	2.051072
4	240000	15.00	0.000315	112402.00	2.794717
4	250000	8.00	0.000180	102078.00	2.032557
4	260000	17.00	0.000353	105322.00	2.745294
4	270000	10.00	0.000251	88840.00	2.062840
4	280000	15.00	0.000368	94300.00	2.191243
4	290000	8.00	0.000191	79909.00	1.639328
4	300000	18.00	0.000364	105128.00	2.443022
4	310000	8.00	0.000188	120579.00	2.260459
4	320000	8.00	0.000216	125597.00	3.134282
4	330000	10.00	0.000236	151182.00	3.940926
4	340000	9.00	0.000196	166492.00	3.507791
4	350000	7.00	0.000181	151689.00	3.194340
4	360000	7.00	0.000186	166928.00	3.850720
4	370000	9.00	0.000212	167154.00	3.666068
4	380000	10.00	0.000162	147368.00	2.885593
4	390000	21.00	0.000388	137803.00	2.969532
4	400000	9.00	0.000202	158026.00	3.690540
4	410000	10.00	0.000256	153238.00	4.177567
4	420000	15.00	0.000350	168440.00	4.543331
4	430000	8.00	0.000174	195902.00	4.567199
4	440000	7.00	0.000192	165922.00	4.078337
4	450000	8.00	0.000199	183992.00	4.580142
4	460000	8.00	0.000190	208746.00	4.302464
4	470000	14.00	0.000234	229321.00	5.470295
4	480000	8.00	0.000210	199475.00	5.379294
4	490000	10.00	0.000266	239623.00	6.374744
4	500000	11.00	0.000238	186026.00	4.691640
4	510000	12.00	0.000298	201041.00	4.919859
4	520000	9.00	0.000218	189501.00	4.317819
4	530000	9.00	0.000205	151069.00	3.144941
4	540000	8.00	0.000219	230280.00	5.777490
4	550000	7.00	0.000164	254817.00	5.943550
4	560000	20.00	0.000433	240510.00	6.015034
4	570000	7.00	0.000178	260619.00	5.542112
4	580000	7.00	0.000177	230100.00	4.817907
4	590000	11.00	0.000239	240188.00	5.613709
4	600000	8.00	0.000202	260924.00	5.703650
4	610000	7.00	0.000179	261019.00	6.318097
4	620000	13.00	0.000295	281111.00	5.626782
4	630000	9.00	0.000227	276200.00	5.939048
4	640000	10.00	0.000227	286193.00	7.329997
4	650000	10.00	0.000213	255659.00	5.345079
4	660000	6.00	0.000165	321451.00	8.061329
4	670000	7.00	0.000170	243797.00	5.014862
4	680000	8.00	0.000173	286228.00	6.904159
4	690000	19.00	0.000396	306680.00	8.027268
4	700000	4.00	0.000134	330990.00	7.337250

## 7 Conclusions

We have fine-tuned the existing decomposition theorem originally proposed by Kao et al. in [19], in the context of maximum weight bipartite matching and applied it to design a revised version of the decomposition algorithm to compute the weight of a maximum weight bipartite matching in  $O(\sqrt{|V|}W'/k(|V|, W'/N))$  time by employing an algorithm designed by Feder and Motwani [10], as base algorithm. We have also analyzed the algorithm by using Hopcroft-Karp algorithm [16] and Alt-Blum-Mehlhorn-Paul algorithm [1] as base algorithms, respectively.

The algorithm performs well especially when the largest edge weight differs by more than one from the second largest edge weight in the current working graph during an invocation of WT-MWBM() in any iteration. Further, we have given a scaling property of the algorithm and a bound of the parameter  $W'$  as  $|E| \leq W' \leq \frac{W}{\text{GCD}(w_1, w_2, \dots, w_{|E|})} \leq W$ , where  $\text{GCD}(w_1, w_2, \dots, w_{|E|})$  denotes the GCD of the positive edges weights  $\{w_1, w_2, \dots, w_{|E|}\}$  of the weighted bipartite graph. The algorithm works well for general  $W$ , but is the best known for  $W' = o(|E| \log(|V|N))$ . The experimental study shows that performance of the modified decomposition algorithm is satisfactory.

## Acknowledgements

The authors are grateful to Dr. Kalpesh Kapoor for his helpful comments and suggestions.

## References

- [1] Helmut Alt, Norbert Blum, Kurt Mehlhorn, and Markus Paul, *Computing a maximum cardinality matching in a bipartite graph in time  $O(n^{1.5} \sqrt{m/\log n})$* , Information Processing Letters **37** (1991), no. 4, 237–240.
- [2] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*, 5<sup>th</sup> ed., North-Holland, NY, USA, 1982.
- [3] ———, *Graph theory*, 1<sup>st</sup> ed., vol. 244, ch. 16 Matchings, p. 419, Springer, 2008.
- [4] Joseph Cheriyan, Torben Hagerup, and Kurt Mehlhorn, *Can a maximum flow be computed in  $o(nm)$  time?*, 17<sup>th</sup> International Colloquium on Automata, Languages and Programming (New York, NY, USA), Springer-Verlag, 1990, pp. 235–248.
- [5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms, third edition*, 3<sup>rd</sup> ed., The MIT Press, 2009.
- [6] Shibsankar Das and Kalpesh Kapoor, *Fine-tuning decomposition theorem for maximum weight bipartite matching*, Theory and Applications of Models of Computation, Lecture Notes in Computer Science, vol. 8402, Springer International Publishing, 2014, pp. 312–322.
- [7] ———, *Approximate parameterized string matching under weighted hamming distance*, AKCE International Journal of Graphs and Combinatorics ((to appear) 2016).
- [8] E. A. Dinic and M. A. Kronrod, *An algorithm for the solution of the assignment problem*, Soviet Mathematics Doklady **10** (1969), 1324–1326.

- [9] Ran Duan and Seth Pettie, *Approximating maximum weight matching in near-linear time*, Annual Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 2010, pp. 673–682.
- [10] Tomás Feder and Rajeev Motwani, *Clique partitions, graph compression and speeding-up algorithms*, Journal of Computer and System Sciences **51** (1995), no. 2, 261–272.
- [11] Michael L. Fredman and Robert Endre Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, Journal of the ACM **34** (1987), no. 3, 596–615.
- [12] Harold N. Gabow, *Scaling algorithms for network problems*, Journal of Computer and System Sciences **31** (1985), no. 2, 148–168.
- [13] Harold N. Gabow and Robert E. Tarjan, *Faster scaling algorithms for network problems*, SIAM Journal on Computing **18** (1989), no. 5, 1013–1036.
- [14] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*, Cambridge University Press, 1934.
- [15] Carmit Hazay, Moshe Lewenstein, and Dina Sokol, *Approximate parameterized matching*, ACM Transactions on Algorithms **3** (2007), no. 3.
- [16] John E. Hopcroft and Richard M. Karp, *An  $n^{5/2}$  algorithm for maximum matchings in bipartite graphs*, SIAM Journal on Computing **2** (1973), no. 4, 225–231.
- [17] M. Iri, *A new method for solving transportation-network problems*, Journal of the Operations Research Society of Japan **3** (1960), 27–87.
- [18] Ming Yang Kao, Tak Wah Lam, Wing Kin Sung, and Hing Fung Ting, *A decomposition theorem for maximum weight bipartite matchings with applications to evolutionary trees*, Annual European Symposium on Algorithms, Springer-Verlag, 1999, pp. 438–449.
- [19] ———, *A decomposition theorem for maximum weight bipartite matchings*, SIAM Journal on Computing **31** (2001), no. 1, 18–26.
- [20] Bernhard Korte and Jens Vygen, *Combinatorial optimization: Theory and algorithms*, 4<sup>th</sup> ed., Springer, 2007.
- [21] H. W. Kuhn, *The hungarian method for the assignment problem*, Naval Research Logistics Quarterly **2** (1955), 83–97.
- [22] James Munkres, *Algorithms for the assignment and transportation problems*, Journal of the Society for Industrial and Applied Mathematics **5** (1957), no. 1, 32–38.
- [23] Piotr Sankowski, *Maximum weight bipartite matching in matrix multiplication time*, Theoretical Computer Science **410** (2009), no. 44, 4480–4488.
- [24] A. Schrijver, *Combinatorial optimization - polyhedra and efficiency*, vol. 24 A, Springer, 2003.
- [25] Douglas B. West, *Introduction to graph theory*, 2<sup>nd</sup> ed., Prentice Hall, September 2000.

## 8 Appendix: Detailed Complexity Analysis of Algorithm 2

Here we give complexity analysis of the Algorithm 2 in general. It is almost similar as done in the paper [19]. We assume that a maximum heap [5] is used to store the distinct edge weights along with the associated edges of  $G$ .

Let the running time of  $\text{WT-MWBM}(G)$  be  $T(|V|, W', N)$  excluding the initialization. Let  $L$  be the set of the heaviest weight edges in  $G$ . So up to the Step 3, construction of  $G_h$  requires  $O(|L| \log |E|)$  time. The Step 4 takes  $O(\sqrt{|V|}|L|/k(|V|, |L|))$  time by using Feder and Motwani's algorithm [10] to compute  $mm(G_h)$ . In Step 5,  $C_h$  can be found in  $O(|L|)$  time from this matching. Let  $L_1$  be the set of edges of  $G$  adjacent to some node  $u$  with  $C_h(u) > 0$ , that is,  $L_1$  consist of edges of  $G$  whose weights reduce in  $G_h^\Delta$ . Let  $l_1 = |L_1|$ . Step 6 updates every edges of  $L_1$  in the heap in  $O(l_1 \log |E|)$  time. Since  $L \subseteq L_1$ , Step 1 to 6 takes  $O(\sqrt{|V|}l_1/k(|V|, l_1))$  time altogether. Let  $l_i = |L_i|$  for  $i = 1, 2, \dots, p \leq N$  and  $h_i = H_1 - H_2$  for  $i$ -th phase of the recursion, where  $L_i$  consists of edges of remaining  $G$  whose weights reduce in  $G_h^\Delta$  on  $i$ -th iteration. Note that,

$$l_1 h_1 + l_2 h_2 + \dots + l_p h_p = W.$$

Let  $l_1 + l_2 + \dots + l_p = W'$ . Observe that if  $h_i = 1$  for all  $i \in [1, p]$ , then  $W' = \sum_{i=1}^p l_i = W$ . Step 7 uses at most  $T(|V|, W'', N'')$  time, where  $W'' (< W')$  is the total weight of  $G_h^\Delta$  and  $N'' (< N)$  is the maximum edge weight of  $G_h^\Delta$ . Hence the recurrence relation for running time is

$$T(|V|, W', N) = O(\sqrt{|V|}l_1/k(|V|, l_1)) + T(|V|, W'', N'') \quad \text{and}$$

$$T(|V|, 0, 0) = 0$$

Therefore,  $T(|V|, W', N)$

$$\begin{aligned} &= O\left(\frac{\sqrt{|V|}l_1}{k(|V|, l_1)}\right) + O\left(\frac{\sqrt{|V|}l_2}{k(|V|, l_2)}\right) + \dots + O\left(\frac{\sqrt{|V|}l_p}{k(|V|, l_p)}\right) \\ &= O\left(\sqrt{|V|}\left(\frac{l_1}{k(|V|, l_1)} + \frac{l_2}{k(|V|, l_2)} + \dots + \frac{l_p}{k(|V|, l_p)}\right)\right) \\ &= O\left(\frac{\sqrt{|V|}}{\log |V|}\left(\log |V|^2 \sum_{i=1}^p l_i - \sum_{i=1}^p l_i \log l_i\right)\right) \end{aligned}$$

Let  $f(x) = x \log x$ . Note that it is a convex function, so by Jensen's inequality<sup>(iv)</sup>,

$$\begin{aligned} \sum_{i=1}^p l_i \log l_i &= \sum_{i=1}^p f(l_i) \geq pf\left(\frac{\sum_{i=1}^p l_i}{p}\right) = pf\left(\frac{W'}{p}\right) \\ &= p \frac{W'}{p} \log \frac{W'}{p} = W' \log \frac{W'}{p} = O\left(W' \log \frac{W'}{N}\right) \end{aligned}$$

This lead to the running time complexity as follows.

$$\begin{aligned} T(|V|, W', N) &= O\left(\frac{\sqrt{|V|}}{\log |V|} \left(W' \log |V|^2 - W' \log \frac{W'}{N}\right)\right) \\ &= O\left(\frac{\sqrt{|V|}W'}{\log |V| / \log \frac{|V|^2}{W'/N}}\right) \\ &= O(\sqrt{|V|}W'/k(|V|, W'/N)). \end{aligned}$$

This is better than the  $O(\sqrt{|V|}W/k(|V|, W/N))$  time as mentioned in [19].

The parameter  $W'$  is smaller than  $W$  which is the total weight of  $G$ , essentially when the heaviest edge weight differs by more than one unit from the second heaviest edge weight in a current working graph during a decomposition in any iteration of the algorithm. In best case the algorithm takes  $O(\sqrt{|V|}|E|/k(|V|, |E|))$  time to compute a maximum weight matching and in worst case  $O(\sqrt{|V|}W/k(|V|, W/N))$ , that is,  $|E| \leq W' \leq W$ . This time complexity bridges a gap between the best known time complexity for computing a Maximum Cardinality Matching (MCM) of unweighted bipartite graph and that of computing a MWBM of a weighted bipartite graph.

However, it is very difficult and challenging to get rid of  $W$  or  $N$  from the complexity. This modified algorithm works well for general  $W$ , but is best known for  $W' = o(|E| \log(|V|N))$ .

---

<sup>(iv)</sup> **Jensen's Inequality** [14]. If  $f(x)$  is a convex function on an interval  $I$  and  $\mu_1, \mu_2, \dots, \mu_n$  are positive weights such that  $\sum_{i=0}^n \mu_i = 1$  then

$$f\left(\sum_0^n \mu_i x_i\right) \leq \sum_{i=0}^n \mu_i f(x_i).$$